

12-05-05

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Richard J. Carter

Application No.: 09/272,810

Filing Date: March 19, 1999

Title: NETWORK SERVER USING LOCAL INFORMATION TO DETECT TIMED-OUT CLIENT REQUESTS



Confirmation No.: 6119

Examiner: L. H. Luu

Group Art Unit: 2141

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on October 3, 2005.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

() (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

() one month	\$120.00
() two months	\$450.00
() three months	\$1020.00
() four months	\$1590.00

() The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of \$500.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EV 568260223US addressed to: Commissioner for Patents, Alexandria, VA 22313-1450
Date of Deposit: December 2, 2005

OR

() I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number _____ on _____

Number of pages:

Typed Name: Gail L. Miller

Signature: Gail L. Miller

Respectfully submitted,

Richard J. Carter

By Jody C. Bishop

Jody C. Bishop

Attorney/Agent for Applicant(s)

Reg. No. 44,034

Date: Dec. 2, 2005

Telephone No.: (214) 855-8007

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Docket No.: 10982056-1
(PATENT)



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Richard J. Carter

Application No.: 09/272,810

Confirmation No.: 6119

Filed: March 19, 1999

Art Unit: 2141

For: NETWORK SERVER USING LOCAL
INFORMATION TO DETECT TIMED-OUT
CLIENT REQUESTS

Examiner: L. H. Luu

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

As required under § 41.37(a), this brief is filed within two months of the Notice of Appeal filed in this case on October 3, 2005, and is in furtherance of said Notice of Appeal.

The fees required under § 41.20(b)(2) are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1206:

- | | |
|-------|---|
| I. | Real Party In Interest |
| II. | Related Appeals and Interferences |
| III. | Status of Claims |
| IV. | Status of Amendments |
| V. | Summary of Claimed Subject Matter |
| VI. | Grounds of Rejection to be Reviewed on Appeal |
| VII. | Argument |
| VIII. | Claims |

IX.	Evidence
X.	Related Proceedings
Appendix A	Claims
Appendix B	Evidence
Appendix C	Related Proceedings

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 31 claims pending in application.

B. Current Status of Claims

1. Claims canceled: None
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-31
4. Claims allowed: None
5. Claims rejected: 1-31

C. Claims On Appeal

The claims on appeal are claims 1-31

IV. STATUS OF AMENDMENTS

The present application was filed March 19, 1999. Applicant filed a Request for Continued Examination (RCE) with an accompanying amendment on November 17, 2004. An Office Action was then mailed March 4, 2005, and Applicant submitted a response on May 11, 2005. A Final Office Action was then mailed on August 3, 2005, from which the present appeal was taken. Thus, Applicant did not file an Amendment in response to the Final Office Action, but instead filed a Notice of Appeal which this brief supports. Accordingly, the claims on appeal are those rejected in the Final Office Action of August 3, 2005, a complete listing of which are provided in Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

According to one claimed embodiment, such as that of independent claim 1, a method comprises establishing a network connection between a server (e.g., server 102 of FIGURES 2-3) and an external client (e.g., client 106 of FIGURES 2-3), where the network connection includes a client-to-server channel (e.g., client-to-server channel 150 of FIGURE 3, and *see* page 6, lines 22-24 of the specification) and a server-to-client channel (e.g., server-to-client channel 152 of FIGURE 3, and *see* page 6, lines 22-24 of the specification). The method further comprises receiving at the server a request from the client for a response by the server (*see e.g.*, page 1, lines 16-20 and page 5, line 26 – page 7, line 15 of the specification). Before preparing a response to the client request, the server examines local server information to determine whether the client-to-server channel of the network connection with the requesting client is still established (*see e.g.*, page 3, lines 17-20 of the specification). And, the server does not prepare the response to the client request if the client-to-server

channel is determined to be no longer established (*see e.g.*, page 3, lines 23-25 of the specification).

In certain embodiments, for example that of claim 2, a state of the server-to-client channel is inferred according to whether the client-to-server channel is still established, and the server does not prepare the response to the client request if the server-to-client channel is inferred to be closed (*see e.g.*, page 3, lines 20-25 of the specification).

In certain embodiments, such as that of claim 3, the server includes a read buffer (e.g., read buffer 124 of FIGURE 2). The client request is read from the read buffer, and the read buffer is then probed to determine whether the client-to-server channel is still established (*see e.g.*, page 8, line 18 – page 9, line 13 of the specification).

In certain embodiments, such as that of claim 4, the server maintains local information about the state of the client-to-server channel (*see e.g.*, page 3, lines 17-20 of the specification). A specific state of the client-to-server channel is determined by examining the local information, and the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state (*see e.g.*, page 10, line 5 – page 11, line 14 of the specification).

In certain embodiments, such as that of claim 6, the state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared, whereby upon determination that the client-to-server channel is no longer established the response preparation is aborted before the response is prepared by the server for sending to the client (*see e.g.*, page 11, line 15 – page 12, line 8 of the specification).

In certain embodiments, such as that of claim 7, the method further comprises, responsive to receiving the request, the server beginning to prepare the response to the requesting client. The method further comprises generating an interrupt on the server when the client-to-server channel is determined to be no longer established, and responsive to the interrupt, the server aborting preparing the response before the response is prepared for sending to the requesting client (*see e.g.*, page 12, line 9 – page 13, line 11 of the specification).

In another claimed embodiment, such as that of independent claim 8, a network server (e.g., server 102 of FIGURES 2-3) comprises a processing unit (e.g., processor 112 of FIGURE 2), and a network interface card (e.g., network interface card 114 of FIGURE 2) for communicatively coupling with a client (e.g., client 106 of FIGURES 2-3) via a communication network (e.g., network 104 of FIGURE 2). The network server further comprises computer memory (e.g., memory 116 of FIGURE 2) programmed to, responsive to a communicative coupling that includes a client-to-server channel (e.g., client-to-server channel 150 of FIGURE 3, and *see* page 6, lines 22-24 of the specification) and a server-to-client channel (e.g., server-to-client channel 152 of FIGURE 3, and *see* page 6, lines 22-24 of the specification) being established with a client and the server receiving from the client a request for a response, cause the processing unit to:

- (a) examine local server information to determine whether the client-to-server channel is still established with the client requesting a response from the server (*see e.g.*, page 3, lines 17-20 of the specification), and

- (b) prepare a response to the requesting client only if the client-to-server channel with the requesting client is first determined to still be established (*see e.g.*, page 3, lines 23-25 of the specification).

In certain embodiments, such as that of claim 9, a state of the server-to-client channel with the requesting client is inferred according to whether the client-to-server channel with the requesting client is still established. The response preparation is aborted if the server-to-client channel with the requesting client is inferred to be closed (*see e.g.*, page 3, lines 20-25 of the specification).

In certain embodiments, such as that of claim 10, the server further comprises a read buffer (e.g., read buffer 124 of FIGURE 2), wherein a client request is read from the read buffer, and wherein the read buffer is probed to determine whether the client-to-server channel is still established (*see e.g.*, page 8, line 18 – page 9, line 13 of the specification).

In certain embodiments, such as that of claim 11, the memory includes local information about a state of the client-to-server channel (*see e.g.*, page 3, lines 17-20 of the specification). A state of the client-to-server channel is determined by examining the local information, and the response preparation is aborted if the local information indicates that the

client-to-server channel is in the specific state (*see e.g.*, page 10, line 5 – page 11, line 14 of the specification).

In certain embodiments, such as that of claim 13, a state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared (*see e.g.*, page 11, line 15 – page 12, line 8 of the specification).

In certain embodiments, such as that of claim 14, the memory is programmed with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel is determined to be no longer established, and wherein a response to a client request is processed until the interrupt is generated (*see e.g.*, page 12, line 9 – page 13, line 11 of the specification).

According to another claimed embodiment, such as that of independent claim 15, a server (e.g., server 102 of FIGURES 2-3) comprises a processing unit (e.g., processor 112 of FIGURE 2). The server further comprises a first means (e.g., queue 122 of FIGURE 2) for maintaining a queue of connections based on connection requests, each connection communicatively coupling the server with an external client (e.g., client 106 of FIGURES 2-3) via a communication network (e.g., network 104 of FIGURE 2), and each connection including a client-to-server channel (e.g., client-to-server channel 150 of FIGURE 3, and *see* page 6, lines 22-24 of the specification) and a server-to-client channel (e.g., server-to-client channel 152 of FIGURE 3, and *see* page 6, lines 22-24 of the specification). The server further comprises a second means (e.g., OS 118 or program 120 of FIGURE 2, and *see* page 7, line 26 – page 8, line 4 and page 8, lines 12-28 of the specification) for accepting connections from the queue, and a third means (e.g., program 120 of FIGURE 2, and *see* page 8, lines 12-17 of the specification) for examining local server information to determine whether the client-to-server channel of a given connection from the queue is still established. The server further comprises a fourth means (e.g., program 120 of FIGURE 2, and *see* page 8, line 12 – page 9, line 13 of the specification) for aborting response preparation if it is determined that the client-to-server channel of the given connection is no longer established.

In another claimed embodiment, such as that of independent claim 16, an article for a server (e.g., server 102 of FIGURES 2-3) including a processing unit (e.g., processor 112 of FIGURE 2) and a network interface card (e.g., network interface card 114 of FIGURE 2) is

provided. The article comprises computer memory (e.g., memory 116 of FIGURE 2). The article further comprises a server program (e.g., server program 120 of FIGURE 2) encoded in the computer memory, the server program commanding (*see* page 8, line 12 – page 9, line 13 of the specification) the processing unit to:

(a) accept network connections for communicatively coupling the server with external clients (e.g., client 106 of FIGURES 2-3) via a communication network (e.g., network 104 of FIGURE 2), each connection having a client-to-server channel (e.g., client-to-server channel 150 of FIGURE 3, and *see* page 6, lines 22-24 of the specification) and a server-to-client channel (e.g., server-to-client channel 152 of FIGURE 3, and *see* page 6, lines 22-24 of the specification),

(b) before a response to a client requesting the response is prepared by the server, examine local server information to determine whether the client-to-server channel of a connection with the requesting client is still established (*see* page 8, line 12 – page 9, line 13 of the specification), and

(c) abort response preparation if the client-to-server channel of the connection with the requesting client is determined to be no longer established (*see* page 8, line 12 – page 9, line 13 of the specification).

In certain embodiments, such as that of claim 17, a state of the server-to-client channel of the connection with the requesting client is inferred according to whether the corresponding client-to-server channel is still established (*see e.g.*, page 3, lines 20-25 of the specification).

In certain embodiments, such as that of claim 18, the memory is further encoded with local information about a state of the connection with the requesting client; wherein the state of the connection with the requesting client is determined by examining the local information (*see e.g.*, page 3, lines 17-20 of the specification); and wherein response preparation is aborted if the local information indicates that the client-to-server channel of the connection with the requesting client is in the specific state (*see e.g.*, page 3, lines 23-25 of the specification).

In certain embodiments, such as that of claim 19, a state of the client-to-server channel of the connection with the requesting client is determined by polling the local

information, the local information being polled concurrently with the server beginning to prepare the client request (*see e.g.*, page 11, line 15 – page 12, line 8 of the specification).

In certain embodiments, such as that of claim 20, the memory is further encoded with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel of the connection with the requesting client is determined to be no longer established, and wherein responsive to the interrupt the server aborts the response preparation before a response is prepared for sending to the requesting client (*see e.g.*, page 12, line 9 – page 13, line 11 of the specification).

According to another claimed embodiment, such as that of independent claim 21, a computer program (e.g., server program 120 of FIGURE 2) executable by a processing unit (e.g., processor 112 of FIGURE 2) is provided. The program comprises instructions stored to computer-readable media. The instructions comprise instructions for commanding a processing unit of a server computer (e.g., server 102 of FIGURES 2-3) to maintain a queue of network connections with external clients (e.g., client 106 of FIGURES 2-3) based on connection requests. The instructions further comprise instructions for commanding the processing unit to accept connections from the queue (*see* page 7, line 26 – page 8, line 4 and page 8, lines 12-28 of the specification), and instructions for commanding the processing unit to examine local server information to determine whether a client-to-server channel of an accepted connection from the queue is still established (*see* page 8, lines 12-17 of the specification). The instructions further comprise instructions for commanding the processing unit to process a client request associated with the accepted connection to prepare a response to the client if the client-to-server channel of the accepted connection is first determined as still established (*see* page 8, line 12 – page 9, line 13 of the specification). The instructions further comprise instructions for commanding the processing unit to forego response preparation for the associated client request if the client-to-server channel of the accepted connection is determined as no longer established (*see* page 8, line 12 – page 9, line 13 of the specification).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-4, 6-11, and 13-21 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,125,401 to Huras et al. (hereinafter “*Huras*”); and

Claims 5, 12, and 22-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *Huras* in view of U.S. Patent No. 6,563,821 to Hong et al. (hereinafter “*Hong*”).

VII. ARGUMENT

Appellant respectfully traverses the outstanding rejections of the pending claims, and requests that the Board reverse the outstanding rejections in light of the remarks contained herein. The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-heading as required by 37 C.F.R. § 41.37(c)(1)(vii).

I. Rejections Under 35 U.S.C. § 102(e) over *Huras*

Claims 1-4, 6-11, and 13-21 are rejected under 35 U.S.C. § 102(e) as being anticipated by *Huras*. To anticipate a claim under 35 U.S.C. § 102, a single reference must teach every element of the claim, *see* M.P.E.P. § 2131. Appellant respectfully submits that *Huras* fails to teach all elements of these claims, and thus fails to anticipate the claims, as discussed further below.

Independent Claim 1

Independent claim 1 recites:

A method comprising:
establishing a network connection between a server and an external client, the network connection including a client-to-server channel and a server-to-client channel;
receiving at the server a request from the client for a response by the server;
before preparing a response to the client request, the server examining local server information to determine whether the client-to-server channel of the network connection with the requesting client is still established; and
the server not preparing the response to the client request if the client-to-server channel is determined to be no longer established. (Emphasis added).

Huras fails to teach all of the above elements of claim 1.

Huras addresses situations in which the client processes executing on a machine terminate abnormally, such as being terminated by an operating system due to an addressing violation, *see e.g.*, col. 1, lines 1-55. At column 1, lines 10-42, *Huras* provides:

In client-server systems, client processes are typically separate from the service provider, and require inter-process communication with the service provider. Once initiated, client processes typically occupy other system resources in addition to communication resources. For example, in a client-server system wherein the service provider is a Database Management System (DBMS), each client process occupies resources (eg, processes, threads, memory, locks on database data, etc.) in the DBMS. The cumulative resources occupied by clients can be significant, especially in systems which support hundreds, or even thousands of client application processes. It is therefore important for the service provider to deallocate these resources promptly after a client process terminates. Accordingly, client processes are usually designed to notify the service provider upon termination.

In situations where a client process terminates abnormally (for example, termination of the client process by the operating system due to an addressing violation), the service provider is not normally notified that the client has terminated. The client process can no longer notify the service provider because the client process has been terminated. Furthermore, although the operating system is often aware of the termination, since usually the operating system is responsible for the termination, the operating system does not normally notify the service provider that the client process has terminated. The service provider, therefore, must be able to detect the abnormal termination of a client process in order to deallocate the system resources previously allocated to the terminated client. The mechanism for detecting abnormal client termination depends on the inter-process communication mechanism utilized by the system.

Thus, *Huras* is concerned with inter-process communication between processes executing on a common machine in order to deallocate resources when a process abnormally terminates.

Accordingly, much of the teachings of *Huras* are directed to client and server processes that are located on a common computer, rather than to a connection between a server and an external client. For instance, *Huras* expressly states in reference to Figure 1 thereof that “Box 100 represents a single machine computer system”. Col. 4, lines 40-41. Thus, the client processes 140 and 150 and server processes 340 and 350 of *Huras* are executing on a common machine. Accordingly, the inter-process communications between the client processes and server processes of *Huras* are not communications between a server and an external client.

The Final Office Action asserts that *Huras* “inherently teaches a client-server system where a terminal and personal computer both have network interface cards that connect to a main computer via a network (figure 1; col. 1 lines 10-55; col. 4 lines 39-65).” Page 5 of the Final Office Action. Indeed, *Huras* does show a client-server system in which a terminal 141 and PC 151 are connected to the single machine computer system 100. However, as discussed further below, *Huras* is not concerned with the connections between the computer system 100 (e.g., “server”) and the external “clients” (terminal 141 and PC 151). Rather, the portions of *Huras* that are relied upon by the Final Office Action address detecting an abnormal termination of client processes 140 and 150, which execute on computer system 100 with the server processes 340 and 350. Accordingly, while a connection between an external client (terminal 141 and PC 151) and a server (computer 100) exists in *Huras*, the teachings of *Huras* are not concerned with such connection but instead address detecting abnormal termination of client processes executing locally on the server computer 100. The Final Office Action confuses this teaching of *Huras* as addressing a connection between the computer 100 and external clients (e.g., terminal 141 and PC 151), but as discussed further herein, the detection of an abnormal termination of a client process on computer 100 in no way addresses whether a connection (e.g., client-to-server channel) between computer 100 and an external client (e.g., terminal 141 or PC 151) exists.

The above confusion in the Final Office Action is the primary basis as to why the rejections based on *Huras* are improper. The Final Office Action properly indicates that a connection between a server (computer 100) and external clients (terminal 141 and PC 151) exists, but then goes on to improperly rely upon the teachings of *Huras* that address detection of abnormal termination of processes on the server computer 100 as somehow addressing determining whether a connection between the server 100 and external clients still exists. Such abnormal termination of processes on the server 100 in no way addresses whether a connection between the server 100 and external clients still exists. For instance, an external client (e.g., terminal 141 or PC 151) may have a connection with server 100 and such connection may remain established even if a client process executing on server 100 abnormally terminates. Some of the confusion in the Final Office Action may stem from the terminology used in *Huras*. For instance, *Huras* refers to “client” and “server” processes. However, the client processes 140 and 150 and server processes 340 and 350 of *Huras* are

executing on a common machine 100, and the client processes 140 and 150 do not represent a connection with a particular external client, as discussed further below.

Independent claim 1 recites “before preparing a response to the client request, the server examining local server information to determine whether the client-to-server channel of the network connection with the requesting client is still established” (emphasis added). *Huras* does not teach determining whether a client-to-server channel of the network connection with an external computer (e.g., terminal 151 or PC 151) is still established. As discussed further below, *Huras* does not address the network connection between the external clients and the computer 100, but is instead focused on detecting whether processes executing on computer 100 have terminated (which may terminate irrespective of whether a client-to-server channel between an external client and computer 100 remains established).

Huras does not teach determining whether a client-to-server channel with an external client is still established. Therefore, *Huras* does not teach performing actions based on such determination, such as the server not preparing the response to the client request if the client-to-server channel is determined to be no longer established (as recited by claim 1).

Figure 1 of *Huras* shows that a terminal 141 and PC 151 are coupled to single machine computer system 100. While machine 100 may be a server for external client computers 141 and 151, *Huras* does not teach handling the connections between such computer 100 and external client computers 141 and 151 in the manner recited by claim 1. *Huras* does not address the connections of external client computers 141 and 151 to the computer 100. Specifically, *Huras* does not address how to handle the preparation of responses to such external client computers in the event that their communication channels with the computer 100 is terminated.

Rather, *Huras* is concerned with the processes executing on computer 100. The client processes 140, 150 do not represent external computers 141 and 151 in *Huras*. That is, client process 140 is not a representation of external computer 141 (or the client-to-server channel) on computer 100, and client process 150 is not a representation of external computer 151 (or the client-to-server channel) on computer 100. Rather, an application on an external client may interact with a client process on computer 100. Various client processes may be

executing on computer 100 for each of the external computers 141, 151. For instance, *Huras* explains as follows:

When a user operating a terminal 141 runs an application which is designed to interact with a service provider incorporating the present invention, the application program establishes the client process 140 running on the computer system 100. ... (Col. 5, lines 10-14)

A similar initialization process takes place for each new client process. For example, client process 150 can be started by an application program, which could be the same application program which established client process 140, or a different application. For example, one application can be a financial data application whereas the other application can be a human resources application, wherein each accesses data from the same DBMS service provider. (Emphasis added). (Col. 6, lines 17-25).

Accordingly, client processes on computer 100 are processes that support a given application on an external computer (such as computers 141, 151). But, the client processes are not representative of a connection (e.g., client-to-server channel) between an external client and the server. For instance, just because a client process terminates does not necessarily mean that the client-to-server communication channel with an external computer no longer persists.

Huras addresses situations in which the client processes executing on machine 100 terminate abnormally, such as being terminated by an operating system due to an addressing violation, *see e.g.*, col. 1, lines 1-55. Because the client processes do not represent a connection of an external client, an abnormal termination of such client processes does not mean that the external client has terminated its communication. Rather, a client process may abnormally terminate (e.g., due to machine 100's OS detecting an addressing violation) while the associated external client remains communicatively coupled to the machine 100. For instance, client process 140 may abnormally terminate, while a communication channel between external terminal 141 and computer 100 persists.

Accordingly, the inter-process interactions between the client processes and server processes on computer 100 of *Huras* do not teach determining whether a client-to-server channel of a network connection with a client is still established, nor does *Huras* teach performing any action (such as the server not preparing the response to the client request)

based on a determination that a client-to-server channel is no longer established (as *Huras* does not teach determining whether such client-to-server channel is established).

The Final Office Action asserts at page 5 thereof that “Huras teaches determining whether the connection of the terminal and the computer system is still established by using semaphores (col. 7 line 57 – col. 8 line 9 and col. 8 lines 59-67).” Appellant respectfully disagrees. Semaphores are well-known for use in coordinating inter-process communications, such as the client and server processes executing on computer 100 of Figure 1 of *Huras*. For instance, a common definition for semaphore is provided by whatis.com (see http://searchopensource.techtarget.com/sDefinition/0,290660,sid39_gci212959,00.html) as follows:

In programming, especially in Unix systems, semaphores are a technique for coordinating or synchronizing activities in which multiple processes compete for the same operating system resources. A semaphore is a value in a designated place in operating system (or kernel) storage that each process can check and then change. Depending on the value that is found, the process can use the resource or will find that it is already in use and must wait for some period before trying again. Semaphores can be binary (0 or 1) or can have additional values. Typically, a process using semaphores checks the value and then, if it using the resource, changes the value to reflect this so that subsequent semaphore users will know to wait.

The above definition is consistent with *Huras*’ usage of the term. *Huras* does not teach using semaphores for determining whether a connection between a server and an external computer system is still established, but instead teaches using the semaphores for indicating the termination of a process on computer system 100. For instance, at column 1, line 56 – column 2, line 3, *Huras* introduces the usage of semaphores for indicating the termination of a process as follows:

Another mechanism for enabling communication between client and server processes involves the utilization of shared memory. In such a system, the client process and the server process communicate by reading and writing to shared memory segments accessible by both. When shared memory is used, operating system mechanisms called semaphores are typically used for controlling client and server access to the shared memory segments by notifying one process when the other process has written data to a shared memory segment. For example, as a client process writes to shared memory, the client process will post (increment) a semaphore, which will in turn notify a waiting process (in this example, the server process) that data is waiting for

it in shared memory. The waiting process will then read the data, completing the data transfer.

Further, column 7, line 57 – column 8, line 9 and column 8, lines 59-67 (the portions of *Huras* cited by the Final Office Action) provide:

Referring now to FIG. 2b, once send semaphore 256 is posted as a result of step 25, the wait function executed at step 9 will be able to carry out its semop () operation, switching the newly posted semaphore back to a state of not posted and ending the wait function. As shown at 50, Server process 350 then reads flag 252 in order to test whether the flag has been set as a valid request (ie. whether it has been set true) by client process 150 at step 20. Test 50 is executed in order to determine whether the send semaphore 256 was posted by client process 150, which will indicate that client process 150 has written data to shared memory segment 250, or whether send semaphore 256 was posted by the operating system, indicating the termination of client process 150.

In order for the result of test 50 to be positive, flag 252 must be set as an invalid request (ie. the flag value is false). This indicates that the send semaphore 256 has been posted by the operating system due to the termination of client process 150, and the Server Process 350 will therefore execute the terminate server routine, as shown at 80, in order to free up system resources. (col. 7, line 57 – col. 8, line 9).

...

If the client process is terminated, either because the application has completed its processing, or because the operating system terminated the client process for some reason, the operating system will post the send semaphore. In this case, the valid request flag will not have been set to true by the client process. After the posting of the send semaphore, the server process will test the flag as discussed and terminate all resources allocated to the client process 150. (col. 8, lines 59-67).

Thus, *Huras* does not teach using semaphores for determining whether a connection between a server and an external computer system is still established, but instead teaches using the semaphores for indicating the termination of the client process 150 on computer system 100. As discussed above, the client process 150 does not represent a connection between computer system 100 and an external client computer, and thus any termination of client process 150 that may be indicated by a semaphore does not indicate a termination of a connection between the computer system 100 and an external client computer.

In view of the above, independent claim 1 is not anticipated by *Huras*. Therefore, Appellant respectfully requests that the rejection of claim 1 be overturned.

Dependent Claim 2

Dependent claim 2 depends from independent claim 1, and thus inherits all limitations of independent claim 1. It is respectfully submitted that dependent claim 2 is allowable at least because of its dependency from independent claim 1 for the reasons discussed above.

Moreover, dependent claim 2 further recites “wherein a state of the server-to-client channel is inferred according to whether the client-to-server channel is still established; and wherein the server does not prepare the response to the client request if the server-to-client channel is inferred to be closed.” *Huras* fails to teach inferring a state of the server-to-client channel according to whether the client-to-server channel is still established. The Final Office Action cites column 1, lines 43-55 and column 4, line 39 – column 8, line 67 of *Huras* as teaching this element of claim 2, *see* page 3 of the Final Office Action. However, nothing in *Huras* teaches inferring a state of the server-to-client channel according to whether the client-to-server channel is still established. The portions of *Huras* cited in the Final Office Action address detection of abnormal termination of a client process. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client. For instance, column 1, lines 40-55 of *Huras* provides:

The mechanism for detecting abnormal client termination depends on the inter-process communication mechanism utilized by the system.

In some systems, this communication is facilitated by means of a communication protocol (e.g., TCPIP, SNA, NETBIOS). Typically in systems using one of these communication protocols, a polling mechanism is used by the service provider to verify the continued existence of each client at regular intervals. There are two primary disadvantages associated with such polling mechanisms. First, performance of the system is affected because CPU time is required to conduct the polling. Furthermore, such CPU time is used even if no client process abnormally terminates. Second, resources allocated to a terminated client process are not deallocated promptly after termination, but remain allocated until that client is next polled.

The above portion of *Huras*, which the Final Office Action cites as teaching the above element of claim 2, does not address a connection (e.g., server-to-client channel or client-to-server channel) between a server and an external client at all, and certainly fails to teach inferring a state of a server-to-client channel according to whether the client-to-server channel is still established. Rather, the above portion of *Huras* addresses an inter-process

communication mechanism for detecting when a client process abnormally terminates. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

Further, as *Huras* fails to teach inferring a state of the server-to-client channel, nothing in *Huras* teaches taking some action based on such inference. Claim 2 recites that “the server does not prepare the response to the client request if the server-to-client channel is inferred to be closed.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to prepare a response if a server-to-client channel with an external client is inferred to be closed.

In view of the above, claim 2 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 2 be overturned.

Dependent Claim 3

Dependent claim 3 depends from independent claim 1, and thus inherits all limitations of independent claim 1. It is respectfully submitted that dependent claim 3 is allowable at least because of its dependency from independent claim 1 for the reasons discussed above.

Moreover, dependent claim 3 further recites “wherein the server includes a read buffer; wherein the client request is read from the read buffer; and wherein the read buffer is then probed to determine whether the client-to-server channel is still established.” *Huras* fails to teach such a read buffer that is probed to determine whether the client-to-server channel is still established. The Final Office Action cites column 1, lines 43-55 and column 4, line 39 – column 8, line 67 of *Huras* as teaching this element of claim 3, *see* page 3 of the Final Office Action. However, nothing in *Huras* teaches probing a read buffer to determine whether the client-to-server channel is still established. The portions of *Huras* cited in the Final Office Action address detection of abnormal termination of a client process. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

The portions of *Huras* relied upon by the Final Office Action as teaching the above element of claim 3, do not address a connection (e.g., server-to-client channel or client-to-server channel) between a server and an external client at all, and certainly fails to teach probing a read buffer on a server to determine whether a client-to-server channel is still established. Rather, the cited portions of *Huras* addresses an inter-process communication mechanism for detecting when a client process abnormally terminates. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

In view of the above, claim 3 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 3 be overturned.

Dependent Claim 4

Dependent claim 4 depends from independent claim 1, and thus inherits all limitations of independent claim 1. It is respectfully submitted that dependent claim 4 is allowable at least because of its dependency from independent claim 1 for the reasons discussed above.

Moreover, dependent claim 4 further recites “wherein the server maintains local information about the state of the client-to-server channel; wherein a specific state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.” *Huras* fails to teach a server that maintains such local information about the state of the client-to-server channel between a server and an external client. The Final Office Action cites column 1, lines 43-55 and column 4, line 39 – column 8, line 67 of *Huras* as teaching this element of claim 4, *see* page 3 of the Final Office Action. However, nothing in *Huras* teaches examining local information maintained on a server to determine a specific state of the client-to-server channel between the server and an external client. The portions of *Huras* cited in the Final Office Action address detection of abnormal termination of a client process. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

The portions of *Huras* relied upon by the Final Office Action as teaching the above element of claim 4, do not address a connection (e.g., server-to-client channel or client-to-server channel) between a server and an external client at all, and certainly fails to teach examining local information maintained on the server to determine a specific state of the client-to-server channel. Rather, the cited portions of *Huras* addresses an inter-process communication mechanism for detecting when a client process abnormally terminates. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

Further, as *Huras* fails to teach determining a specific state of the client-to-server channel, nothing in *Huras* teaches taking some action based on such determination. Claim 4 recites that “the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to abort preparation of a response to an external client if a client-to-server channel is determined to be in a specific state.

In view of the above, claim 4 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 4 be overturned.

Dependent Claim 6

Dependent claim 6 depends from independent claim 1, and thus inherits all limitations of independent claim 1. It is respectfully submitted that dependent claim 6 is allowable at least because of its dependency from independent claim 1 for the reasons discussed above.

Moreover, dependent claim 6 further recites “wherein the state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared, whereby upon determination that the client-to-server channel is no longer established the response preparation is aborted before the response is prepared by the server for sending to the client.” *Huras* fails to teach polling local information to determine the state of the client-to-server channel between a server and an external client. The Final Office

Action cites column 1, lines 43-55 and column 4, line 39 – column 8, line 67 of *Huras* as teaching this element of claim 6, *see* page 3 of the Final Office Action. However, nothing in *Huras* teaches polling local information on a server to determine the state of the client-to-server channel between the server and an external client. The portions of *Huras* cited in the Final Office Action address detection of abnormal termination of a client process. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

The portions of *Huras* relied upon by the Final Office Action as teaching the above element of claim 6, do not address a connection (e.g., server-to-client channel or client-to-server channel) between a server and an external client at all, and certainly fails to teach polling local information on the server to determine a state of the client-to-server channel. Rather, the cited portions of *Huras* addresses an inter-process communication mechanism for detecting when a client process abnormally terminates. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

Further, as *Huras* fails to teach determining a state of the client-to-server channel, nothing in *Huras* teaches taking some action based on such determination. Claim 6 recites that “upon determination that the client-to-server channel is no longer established the response preparation is aborted before the response is prepared by the server for sending to the client.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to abort preparation of a response to an external client if a client-to-server channel is determined to no longer be established.

In view of the above, claim 6 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 6 be overturned.

Dependent Claim 7

Dependent claim 7 depends from independent claim 1, and thus inherits all limitations of independent claim 1. It is respectfully submitted that dependent claim 7 is allowable at least because of its dependency from independent claim 1 for the reasons discussed above.

Moreover, dependent claim 7 further recites:

The method of claim 1, further comprising:
responsive to receiving said request, the server beginning to prepare the response to the requesting client;
generating an interrupt on the server when the client-to-server channel is determined to be no longer established; and
responsive to said interrupt, the server aborting preparing the response before the response is prepared for sending to the requesting client. (Emphasis added).

Huras fails to teach generating an interrupt on a server when a client-to-server channel between a server and an external client is determined to no longer be established. The Final Office Action cites column 1, lines 43-55 and column 4, line 39 – column 8, line 67 of *Huras* as teaching this element of claim 7, *see* page 3 of the Final Office Action. However, nothing in *Huras* teaches generating an interrupt on a server when the client-to-server channel between the server and an external client is determined to no longer be established. The portions of *Huras* cited in the Final Office Action address detection of abnormal termination of a client process. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

The portions of *Huras* relied upon by the Final Office Action as teaching the above element of claim 7, do not address a connection (e.g., server-to-client channel or client-to-server channel) between a server and an external client at all, and certainly fails to teach generating an interrupt when the client-to-server channel between the server and an external client is determined to no longer be established. Rather, the cited portions of *Huras* addresses an inter-process communication mechanism for detecting when a client process abnormally terminates. As discussed above with claim 1, such client processes execute on a common computer 100 with server processes, and the client processes do not represent a connection with an external client.

Further, as *Huras* fails to teach generating such an interrupt, nothing in *Huras* teaches taking some action responsive to the interrupt. Claim 7 recites that “responsive to said interrupt, the server aborting preparing the response before the response is prepared for sending to the requesting client.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address aborting preparation of a response before the response is prepared for sending to a requesting external client in response to an interrupt that indicates a client-to-server channel is no longer established.

In view of the above, claim 7 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 7 be overturned.

Independent Claim 8

Independent claim 8 recites a network server comprising:

a network interface card for communicatively coupling with a client via a communication network; and

computer memory programmed to, responsive to a communicative coupling that includes a client-to-server channel and a server-to-client channel being established with a client and the server receiving from the client a request for a response, cause the processing unit to

(a) examine local server information to determine whether the client-to-server channel is still established with the client requesting a response from the server, and

(b) prepare a response to the requesting client only if the client-to-server channel with the requesting client is first determined to still be established. (Emphasis added).

As described above with claim 1, the inter-process interactions between the client processes and server processes on computer 100 of *Huras* do not teach determining whether a client-to-server channel of a network connection with a client is still established, and therefore *Huras* does not teach performing any action based on a determination of whether the client-to-server channel is still established, such as preparing a response to a requesting client only if the client-to-server channel with the requesting client is first determined to still be established. While *Huras* is concerned with determining whether a client process executing on computer 100 has terminated, *Huras* does not teach determining whether a

client-to-server channel between computer 100 and an external computer remains established. Again, merely because a client process terminates does not provide a determination in *Huras* as to whether a client-to-server channel with an external computer has terminated (e.g., the client process may terminate due to an addressing violation encountered by the operating system of computer 100).

Thus, independent claim 8 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 8 be overturned.

Dependent Claim 9

Dependent claim 9 depends from independent claim 8, and thus inherits all limitations of independent claim 8. It is respectfully submitted that dependent claim 9 is allowable at least because of its dependency from independent claim 8 for the reasons discussed above.

Moreover, dependent claim 9 further recites “wherein a state of the server-to-client channel with the requesting client is inferred according to whether the client-to-server channel with the requesting client is still established; and wherein the response preparation is aborted if the server-to-client channel with the requesting client is inferred to be closed.” *Huras* fails to teach inferring a state of the server-to-client channel with a requesting external client according to whether the client-to-server channel is still established. As discussed above with dependent claim 2, nothing in *Huras* teaches inferring a state of the server-to-client channel according to whether the client-to-server channel is still established.

Further, as *Huras* fails to teach inferring a state of the server-to-client channel, nothing in *Huras* teaches taking some action based on such inference. Claim 9 recites that “the response preparation is aborted if the server-to-client channel with the requesting client is inferred to be closed.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to abort preparation of a response if a server-to-client channel with an external client is inferred to be closed.

In view of the above, claim 9 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 9 be overturned.

Dependent Claim 10

Dependent claim 10 depends from claim 9, which depends from independent claim 8, and thus claim 10 inherits all limitations of claims 8 and 9. It is respectfully submitted that dependent claim 10 is allowable at least because of its dependency from claims 8 and 9 for the reasons discussed above.

Moreover, dependent claim 10 further recites that the server further comprises “a read buffer; wherein a client request is read from the read buffer; and wherein the read buffer is probed to determine whether the client-to-server channel is still established.” *Huras* fails to teach such a read buffer that is probed to determine whether the client-to-server channel is still established. As discussed above with claim 3, nothing in *Huras* teaches probing a read buffer to determine whether the client-to-server channel is still established.

In view of the above, claim 10 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 10 be overturned.

Dependent Claim 11

Dependent claim 11 depends from independent claim 8, and thus inherits all limitations of independent claim 8. It is respectfully submitted that dependent claim 11 is allowable at least because of its dependency from independent claim 8 for the reasons discussed above.

Moreover, dependent claim 11 further recites “wherein the memory includes local information about a state of the client-to-server channel; wherein a state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.” *Huras* fails to teach a server that maintains such local information about the state of the client-to-server channel between a server and an external client. As discussed above with claim 4, nothing in *Huras* teaches examining local information maintained on a server to determine a specific state of the client-to-server channel between the server and an external client.

Further, as *Huras* fails to teach determining a specific state of the client-to-server channel, nothing in *Huras* teaches taking some action based on such determination. Claim 11 recites that “the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to abort preparation of a response to an external client if a client-to-server channel is determined to be in a specific state.

In view of the above, claim 11 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 11 be overturned.

Dependent Claim 13

Dependent claim 13 depends from independent claim 8, and thus inherits all limitations of independent claim 8. It is respectfully submitted that dependent claim 13 is allowable at least because of its dependency from independent claim 8 for the reasons discussed above.

Moreover, dependent claim 13 further recites “wherein a state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared.” *Huras* fails to teach polling local information to determine the state of the client-to-server channel between a server and an external client. As discussed above with claim 6, nothing in *Huras* teaches polling local information on a server to determine the state of the client-to-server channel between the server and an external client.

In view of the above, claim 13 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 13 be overturned.

Dependent Claim 14

Dependent claim 14 depends from independent claim 8, and thus inherits all limitations of independent claim 8. It is respectfully submitted that dependent claim 14 is

allowable at least because of its dependency from independent claim 8 for the reasons discussed above.

Moreover, dependent claim 14 further recites “wherein the memory is programmed with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel is determined to be no longer established, and wherein a response to a client request is processed until the interrupt is generated.” (Emphasis added).

Huras fails to teach generating an interrupt on a server when a client-to-server channel between a server and an external client is determined to no longer be established. As discussed above with claim 7, nothing in *Huras* teaches generating an interrupt on a server when the client-to-server channel between the server and an external client is determined to no longer be established.

In view of the above, claim 14 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 14 be overturned.

Independent Claim 15

Independent claim 15 recites:

A server comprising:
a processing unit;
first means for maintaining a queue of connections based on connection requests, each connection communicatively coupling the server with an external client via a communication network, and each connection including a client-to-server channel and a server-to-client channel;
second means for accepting connections from the queue;
third means for examining local server information to determine whether the client-to-server channel of a given connection from the queue is still established; and
fourth means for aborting response preparation if it is determined that the client-to-server channel of the given connection is no longer established.
(Emphasis added).

Huras fails to teach all of the above elements of claim 15.

As discussed above with claim 1, much of the teachings of *Huras* are directed to client and server processes that are located on a common computer, rather than to a connection between a server and an external client. *Huras* does show, in Figure 1 thereof, a client-server system in which a terminal 141 and PC 151 are connected to the single machine computer system 100. However, as discussed further below, independent claim 15 recites a server comprising “third means for examining local server information to determine whether the client-to-server channel of a given connection from the queue is still established” (emphasis added). *Huras* does not teach determining whether a client-to-server channel of a given network connection with an external computer (e.g., terminal 151 or PC 151) is still established. As discussed above with claim 1, *Huras* does not address the network connection between the external clients and the computer 100, but is instead focused on detecting whether processes executing on computer 100 have terminated (which may terminate irrespective of whether a client-to-server channel between an external client and computer 100 remains established).

Huras does not teach determining whether a client-to-server channel with an external client is still established. Therefore, *Huras* does not teach performing actions based on such determination, such as a “fourth means for aborting response preparation if it is determined that the client-to-server channel of the given connection is no longer established” (emphasis added), as recited by claim 15.

Figure 1 of *Huras* shows that a terminal 141 and PC 151 are coupled to single machine computer system 100. While machine 100 may be a server for external client computers 141 and 151, *Huras* does not teach handling the connections between such computer 100 and external client computers 141 and 151 in the manner recited by claim 15. *Huras* does not address the connections of external client computers 141 and 151 to the computer 100. Specifically, *Huras* does not address how to handle the preparation of responses to such external client computers in the event that their communication channels with the computer 100 is terminated.

Rather, *Huras* is concerned with the processes executing on computer 100. As discussed above with claim 1, the client processes 140, 150 do not represent external computers 141 and 151 in *Huras*. The client processes on computer 100 are processes that

support a given application on an external computer (such as computers 141, 151), but the client processes are not representative of a connection (e.g., client-to-server channel) between an external client and the server. For instance, just because a client process terminates does not necessarily mean that the client-to-server communication channel with an external computer no longer persists.

Accordingly, the inter-process interactions between the client processes and server processes on computer 100 of *Huras* do not teach determining whether a client-to-server channel of a network connection with a client is still established, nor does *Huras* teach performing any action (such as aborting response preparation) based on a determination that a client-to-server channel is no longer established (as *Huras* does not teach determining whether such client-to-server channel is established).

Thus, independent claim 15 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 15 be overturned.

Independent Claim 16

Independent claim 16 recites:

An article for a server including a processing unit and a network interface card, the article comprising:
computer memory; and
a server program encoded in the computer memory, the server program commanding the processing unit to
(a) accept network connections for communicatively coupling the server with external clients via a communication network, each connection having a client-to-server channel and a server-to-client channel,
(b) before a response to a client requesting the response is prepared by the server, examine local server information to determine whether the client-to-server channel of a connection with the requesting client is still established, and
(c) abort response preparation if the client-to-server channel of the connection with the requesting client is determined to be no longer established. (Emphasis added).

Huras fails to teach all of the above elements of claim 16.

As discussed above with claim 1, much of the teachings of *Huras* are directed to client and server processes that are located on a common computer, rather than to a

connection between a server and an external client. *Huras* does show, in Figure 1 thereof, a client-server system in which a terminal 141 and PC 151 are connected to the single machine computer system 100. However, as discussed further below, independent claim 16 recites a server program commanding the processing unit to “(b) before a response to a client requesting the response is prepared by the server, examine local server information to determine whether the client-to-server channel of a connection with the requesting client is still established” (emphasis added). *Huras* does not teach determining whether a client-to-server channel of a given network connection with an external computer (e.g., terminal 151 or PC 151) is still established. As discussed above with claim 1, *Huras* does not address the network connection between the external clients and the computer 100, but is instead focused on detecting whether processes executing on computer 100 have terminated (which may terminate irrespective of whether a client-to-server channel between an external client and computer 100 remains established).

Huras does not teach determining whether a client-to-server channel with an external client is still established. Therefore, *Huras* does not teach performing actions based on such determination, such as “(c) abort response preparation if the client-to-server channel of the connection with the requesting client is determined to be no longer established” (emphasis added), as recited by claim 16.

Figure 1 of *Huras* shows that a terminal 141 and PC 151 are coupled to single machine computer system 100. While machine 100 may be a server for external client computers 141 and 151, *Huras* does not teach handling the connections between such computer 100 and external client computers 141 and 151 in the manner recited by claim 16. *Huras* does not address the connections of external client computers 141 and 151 to the computer 100. Specifically, *Huras* does not address how to handle the preparation of responses to such external client computers in the event that their communication channels with the computer 100 is terminated.

Rather, *Huras* is concerned with the processes executing on computer 100. As discussed above with claim 1, the client processes 140, 150 do not represent external computers 141 and 151 in *Huras*. The client processes on computer 100 are processes that support a given application on an external computer (such as computers 141, 151), but the

client processes are not representative of a connection (e.g., client-to-server channel) between an external client and the server. For instance, just because a client process terminates does not necessarily mean that the client-to-server communication channel with an external computer no longer persists.

Accordingly, the inter-process interactions between the client processes and server processes on computer 100 of *Huras* do not teach determining whether a client-to-server channel of a network connection with a client is still established, nor does *Huras* teach performing any action (such as aborting response preparation) based on a determination that a client-to-server channel is no longer established (as *Huras* does not teach determining whether such client-to-server channel is established).

Thus, independent claim 16 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 16 be overturned.

Dependent Claim 17

Dependent claim 17 depends from independent claim 16, and thus inherits all limitations of independent claim 16. It is respectfully submitted that dependent claim 17 is allowable at least because of its dependency from independent claim 16 for the reasons discussed above.

Moreover, dependent claim 17 further recites “wherein a state of the server-to-client channel of the connection with the requesting client is inferred according to whether the corresponding client-to-server channel is still established.” *Huras* fails to teach inferring a state of the server-to-client channel with a requesting external client according to whether the client-to-server channel is still established. As discussed above with dependent claim 2, nothing in *Huras* teaches inferring a state of the server-to-client channel according to whether the client-to-server channel is still established.

In view of the above, claim 17 is not anticipated by *Huras* as *Huras* fails to teach all elements of claim 17. Therefore, Appellant respectfully requests that the rejection of claim 17 be overturned.

Dependent Claim 18

Dependent claim 18 depends from independent claim 16, and thus inherits all limitations of independent claim 16. It is respectfully submitted that dependent claim 18 is allowable at least because of its dependency from independent claim 16 for the reasons discussed above.

Moreover, dependent claim 18 further recites “wherein the memory is further encoded with local information about a state of the connection with the requesting client; wherein the state of the connection with the requesting client is determined by examining the local information; and wherein response preparation is aborted if the local information indicates that the client-to-server channel of the connection with the requesting client is in the specific state.” *Huras* fails to teach a server that maintains such local information about the state of the connection with an external client. As discussed above with claim 4, nothing in *Huras* teaches examining local information maintained on a server to determine a specific state of the client-to-server channel between the server and an external client.

Further, as *Huras* fails to teach determining a specific state of the client-to-server channel, nothing in *Huras* teaches taking some action based on such determination. Claim 18 recites that “response preparation is aborted if the local information indicates that the client-to-server channel of the connection with the requesting client is in the specific state.” *Huras* addresses inter-process communication between processes executing on a common computer 100, and does not address whether to abort preparation of a response to an external client if a client-to-server channel is determined to be in a specific state.

In view of the above, claim 18 is not anticipated by *Huras* as *Huras* fails to teach all elements of claim 18. Therefore, Appellant respectfully requests that the rejection of claim 18 be overturned.

Dependent Claim 19

Dependent claim 19 depends from independent claim 16, and thus inherits all limitations of independent claim 16. It is respectfully submitted that dependent claim 19 is

allowable at least because of its dependency from independent claim 16 for the reasons discussed above.

Moreover, dependent claim 19 further recites “wherein a state of the client-to-server channel of the connection with the requesting client is determined by polling the local information, the local information being polled concurrently with the server beginning to prepare the client request.” *Huras* fails to teach polling local information to determine the state of the client-to-server channel between a server and an external client. As discussed above with claim 6, nothing in *Huras* teaches polling local information on a server to determine the state of the client-to-server channel between the server and an external client.

In view of the above, claim 19 is not anticipated by *Huras* as *Huras* fails to teach all elements of claim 19. Therefore, Appellant respectfully requests that the rejection of claim 19 be overturned.

Dependent Claim 20

Dependent claim 20 depends from independent claim 16, and thus inherits all limitations of independent claim 16. It is respectfully submitted that dependent claim 20 is allowable at least because of its dependency from independent claim 16 for the reasons discussed above.

Moreover, dependent claim 20 further recites “wherein the memory is further encoded with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel of the connection with the requesting client is determined to be no longer established, and wherein responsive to the interrupt the server aborts the response preparation before a response is prepared for sending to the requesting client.” (Emphasis added).

Huras fails to teach generating an interrupt on a server when a client-to-server channel between a server and an external client is determined to no longer be established. As discussed above with claim 7, nothing in *Huras* teaches generating an interrupt on a server when the client-to-server channel between the server and an external client is determined to no longer be established.

In view of the above, claim 20 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 20 be overturned.

Independent Claim 21

Independent claim 21 recites:

A computer program executable by a processing unit, the program comprising instructions stored to computer-readable media, the instructions comprising:

instructions for commanding a processing unit of a server computer to maintain a queue of network connections with external clients based on connection requests;

instructions for commanding the processing unit to accept connections from the queue;

instructions for commanding the processing unit to examine local server information to determine whether a client-to-server channel of an accepted connection from the queue is still established;

instructions for commanding the processing unit to process a client request associated with the accepted connection to prepare a response to the client if the client-to-server channel of the accepted connection is first determined as still established; and

instructions for commanding the processing unit to forego response preparation for the associated client request if the client-to-server channel of the accepted connection is determined as no longer established. (Emphasis added).

Huras fails to teach all of the above elements of claim 21.

As discussed above with claim 1, much of the teachings of *Huras* are directed to client and server processes that are located on a common computer, rather than to a connection between a server and an external client. *Huras* does show, in Figure 1 thereof, a client-server system in which a terminal 141 and PC 151 are connected to the single machine computer system 100. However, as discussed further below, independent claim 21 recites “instructions for commanding the processing unit to examine local server information to determine whether a client-to-server channel of an accepted connection from the queue is still established” (emphasis added). *Huras* does not teach determining whether a client-to-server channel of a given network connection with an external computer (e.g., terminal 151 or PC 151) is still established. As discussed above with claim 1, *Huras* does not address the

network connection between the external clients and the computer 100, but is instead focused on detecting whether processes executing on computer 100 have terminated (which may terminate irrespective of whether a client-to-server channel between an external client and computer 100 remains established).

Huras does not teach determining whether a client-to-server channel with an external client is still established. Therefore, *Huras* does not teach performing actions based on such determination, such as “instructions for commanding the processing unit to process a client request associated with the accepted connection to prepare a response to the client if the client-to-server channel of the accepted connection is first determined as still established; and instructions for commanding the processing unit to forego response preparation for the associated client request if the client-to-server channel of the accepted connection is determined as no longer established” (emphasis added), as recited by claim 21.

Figure 1 of *Huras* shows that a terminal 141 and PC 151 are coupled to single machine computer system 100. While machine 100 may be a server for external client computers 141 and 151, *Huras* does not teach handling the connections between such computer 100 and external client computers 141 and 151 in the manner recited by claim 21. *Huras* does not address the connections of external client computers 141 and 151 to the computer 100. Specifically, *Huras* does not address how to handle the preparation of responses to such external client computers in the event that their communication channels with the computer 100 is terminated.

Rather, *Huras* is concerned with the processes executing on computer 100. As discussed above with claim 1, the client processes 140, 150 do not represent external computers 141 and 151 in *Huras*. The client processes on computer 100 are processes that support a given application on an external computer (such as computers 141, 151), but the client processes are not representative of a connection (e.g., client-to-server channel) between an external client and the server. For instance, just because a client process terminates does not necessarily mean that the client-to-server communication channel with an external computer no longer persists.

Accordingly, the inter-process interactions between the client processes and server processes on computer 100 of *Huras* do not teach determining whether a client-to-server

channel of a network connection with a client is still established, nor does *Huras* teach performing any action (such as foregoing response preparation) based on a determination that a client-to-server channel is no longer established (as *Huras* does not teach determining whether such client-to-server channel is established).

Thus, independent claim 21 is not anticipated by *Huras* at least for the above reasons. Therefore, Appellant respectfully requests that the rejection of claim 21 be overturned.

II. Rejections Under 35 U.S.C. § 103(a) over *Huras* in view of *Hong*

Claims 5, 12, and 22-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *Huras* in view of *Hong*. Dependent claims 5, 12, and 22-31 each depend either directly or indirectly from one of independent claims 1, 8, 15, 16, and 21, and thus inherit all limitations of the respective independent claim from which they depend. Further, the Final Office Action does not rely on *Hong* to cure the deficiencies of *Huras* identified above for claims 1, 8, 15, 16, and 21, as *Hong* does not cure such deficiencies. It is respectfully submitted that dependent claims 5, 12, and 22-31 are allowable not only because of their dependency from their respective independent claims for the reasons discussed above, but also in view of their novel claim features (which both narrow the scope of the particular claims and compel a broader interpretation of the respective base claim from which they depend).

VIII. CLAIMS

A copy of the claims involved in the present appeal is attached hereto as Appendix A. As indicated above, the claims in Appendix A include the amendments filed by Applicant on May 11, 2005.

IX. EVIDENCE

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

X. RELATED PROCEEDINGS

As noted in Appendix C hereto, no related proceedings are referenced in II. above, nor are copies of decisions in related proceedings provided.

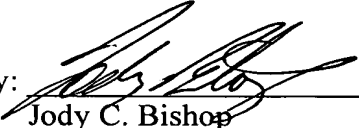
I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EV 568260223 US in an envelope addressed to: M/S Appeal Brief - Patents, Commissioner for Patents, Alexandria, VA 22313.

Date of Deposit: December 2, 2005

Typed Name: Gail L. Miller

Signature: Gail L. Miller

Respectfully submitted,

By: 
Jody C. Bishop
Attorney/Agent for Applicant(s)
Reg. No. 44,034
Date: December 2, 2005
Telephone No. (214) 855-8007

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/272,810

1. A method comprising:
establishing a network connection between a server and an external client, the network connection including a client-to-server channel and a server-to-client channel;
receiving at the server a request from the client for a response by the server;
before preparing a response to the client request, the server examining local server information to determine whether the client-to-server channel of the network connection with the requesting client is still established; and
the server not preparing the response to the client request if the client-to-server channel is determined to be no longer established.
2. The method of claim 1, wherein a state of the server-to-client channel is inferred according to whether the client-to-server channel is still established; and wherein the the server does not prepare the response to the client request if the server-to-client channel is inferred to be closed.
3. The method of claim 1, wherein the server includes a read buffer; wherein the client request is read from the read buffer; and wherein the read buffer is then probed to determine whether the client-to-server channel is still established.
4. The method of claim 1, wherein the server maintains local information about the state of the client-to-server channel; wherein a specific state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.
5. The method of claim 4, wherein the client-to-server channel is determined to be no longer established if the local information indicates that the client-to-server channel is in a "CLOSE_WAIT" state.

6. The method of claim 1, wherein the state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared, whereby upon determination that the client-to-server channel is no longer established the response preparation is aborted before the response is prepared by the server for sending to the client.

7. The method of claim 1, further comprising:
responsive to receiving said request, the server beginning to prepare the response to the requesting client;
generating an interrupt on the server when the client-to-server channel is determined to be no longer established; and
responsive to said interrupt, the server aborting preparing the response before the response is prepared for sending to the requesting client.

8. A network server comprising:
a processing unit;
a network interface card for communicatively coupling with a client via a communication network; and
computer memory programmed to, responsive to a communicative coupling that includes a client-to-server channel and a server-to-client channel being established with a client and the server receiving from the client a request for a response, cause the processing unit to

(a) examine local server information to determine whether the client-to-server channel is still established with the client requesting a response from the server, and
(b) prepare a response to the requesting client only if the client-to-server channel with the requesting client is first determined to still be established.

9. The server of claim 8, wherein a state of the server-to-client channel with the requesting client is inferred according to whether the client-to-server channel with the requesting client is still established; and wherein the response preparation is aborted if the server-to-client channel with the requesting client is inferred to be closed.

10. The server of claim 9, further comprising a read buffer; wherein a client request is read from the read buffer; and wherein the read buffer is probed to determine whether the client-to-server channel is still established

11. The server of claim 8, wherein the memory includes local information about a state of the client-to-server channel; wherein a state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.

12. The server of claim 11, wherein the client-to-server channel is determined to be no longer established if the local information indicates that the client-to-server channel is in a "CLOSE_WAIT" state.

13. The server of claim 8, wherein a state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared.

14. The server of claim 8, wherein the memory is programmed with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel is determined to be no longer established, and wherein a response to a client request is processed until the interrupt is generated.

15. A server comprising:
a processing unit;
first means for maintaining a queue of connections based on connection requests, each connection communicatively coupling the server with an external client via a communication network, and each connection including a client-to-server channel and a server-to-client channel;
second means for accepting connections from the queue;
third means for examining local server information to determine whether the client-to-server channel of a given connection from the queue is still established; and
fourth means for aborting response preparation if it is determined that the client-to-server channel of the given connection is no longer established.

16. An article for a server including a processing unit and a network interface card, the article comprising:

computer memory; and

a server program encoded in the computer memory, the server program commanding the processing unit to

(a) accept network connections for communicatively coupling the server with external clients via a communication network, each connection having a client-to-server channel and a server-to-client channel,

(b) before a response to a client requesting the response is prepared by the server, examine local server information to determine whether the client-to-server channel of a connection with the requesting client is still established, and

(c) abort response preparation if the client-to-server channel of the connection with the requesting client is determined to be no longer established.

17. The article of claim 16, wherein a state of the server-to-client channel of the connection with the requesting client is inferred according to whether the corresponding client-to-server channel is still established.

18. The article of claim 16, wherein the memory is further encoded with local information about a state of the connection with the requesting client; wherein the state of the connection with the requesting client is determined by examining the local information; and wherein response preparation is aborted if the local information indicates that the client-to-server channel of the connection with the requesting client is in the specific state.

19. The article of claim 16, wherein a state of the client-to-server channel of the connection with the requesting client is determined by polling the local information, the local information being polled concurrently with the server beginning to prepare the client request.

20. The article of claim 16, wherein the memory is further encoded with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel of the connection with the requesting client is determined to be no longer established, and wherein responsive to the interrupt the server aborts the response preparation before a response is prepared for sending to the requesting client.

21. A computer program executable by a processing unit, the program comprising instructions stored to computer-readable media, the instructions comprising:

instructions for commanding a processing unit of a server computer to maintain a queue of network connections with external clients based on connection requests;

instructions for commanding the processing unit to accept connections from the queue;

instructions for commanding the processing unit to examine local server information to determine whether a client-to-server channel of an accepted connection from the queue is still established;

instructions for commanding the processing unit to process a client request associated with the accepted connection to prepare a response to the client if the client-to-server channel of the accepted connection is first determined as still established; and

instructions for commanding the processing unit to forego response preparation for the associated client request if the client-to-server channel of the accepted connection is determined as no longer established.

22. The method of claim 1 wherein said server is a web server.

23. The method of claim 22 wherein said response is a web page requested by the requesting client.

24. The network server of claim 8 wherein said network server is a web server.

25. The network server of claim 24 wherein said requested response is a web page.

26. The server of claim 15 wherein said communication network is the Internet.

27. The server of claim 15 wherein said server is a web server for serving web pages to clients via said communication network.

28. The article of claim 16 wherein the communication network is one selected from the group consisting of:

a wide area network, a local area network, the Internet.

29. The article of claim 16 wherein the server is a web server.
30. The computer program of claim 21 wherein said server computer is a web server.
31. The computer program of claim 30 wherein said client request is a request for a web page.

APPENDIX B

Evidence

None.

APPENDIX C

Related Proceedings

None.